
Defeaturing CAD Models Using a Geometry-Based Size Field and Facet-Based Reduction Operators

William Roshan Quadros and Steven J. Owen

*Sandia National Laboratories, Albuquerque, New Mexico, U.S.A
{wrquadros,sjowen}@sandia.gov

Abstract. We propose a method to automatically defeature a CAD model by detecting irrelevant features using a geometry-based size field and a method to remove the irrelevant features via facet-based operations on a discrete representation. A discrete B-Rep model is first created by obtaining a faceted representation of the CAD entities. The candidate facet entities are then marked for reduction by using a geometry-based size field. This is accomplished by estimating local mesh sizes based on geometric criteria. If the field value at a facet entity goes below a user specified threshold value then it is identified as an irrelevant feature and is marked for reduction. The reduction of marked facet entities is primarily performed using an edge collapse operator. Care is taken to retain a valid geometry and topology of the discrete model throughout the procedure. The original model is not altered as the defeaturing is performed on a separate discrete model. Associativity between the entities of the discrete model and that of original CAD model is maintained in order to decode the attributes and boundary conditions applied on the original CAD entities onto the mesh via the entities of the discrete model. Example models are presented to illustrate the effectiveness of the proposed approach.

Keywords: defeaturing, feature suppression, CAD simplification, facet reduction.

1 Introduction

CAD technology has evolved significantly in recent decades, facilitating complex and detailed modeling in the early design stages of computational simulation. This has brought new challenges in the pre analysis stages including the defeaturing of irrelevant or unwanted features prior to mesh generation. This process frequently requires extensive user interaction to eliminate unwanted features and misalignments between parts. Therefore, automatic defeaturing procedures that reduce user time and increase the success ratio of mesh generation are needed. Defeaturing also reduces the degrees of freedom, thereby decreasing the analysis time and memory usage.

* Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Defeaturing, sometimes known as feature suppression or simplification, is intended to address a wide range of characteristics, typically included for design purposes, that may not be relevant or desired for finite element (FE) simulation. Some of these geometric features often present in single body solid models may include small features such as holes, pegs, slots, fillets, chamfers, slender/sliver surfaces and thin-wall regions. Retaining the features described here may result in smaller mesh sizes at these localized regions which can lead to mesh size transition and mesh quality issues. This might lead to ill-conditioning of the FE model and using excessive computing power may not help. In addition, many practical hex meshing algorithms such as sweeping and mapping, which are sensitive to topological features can often fail while accounting for these anomalies.

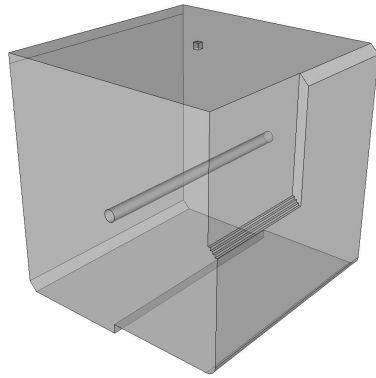


Fig. 1a. CAD model with chamfer, hole, steps, pegs, fillet, and imprint

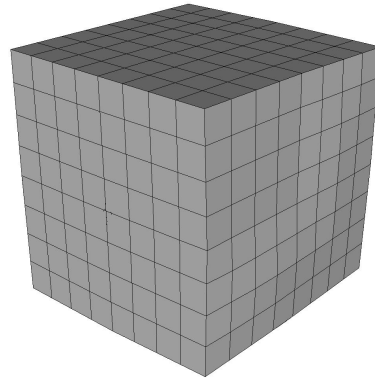


Fig. 1b. Hex mesh of the defeatured model

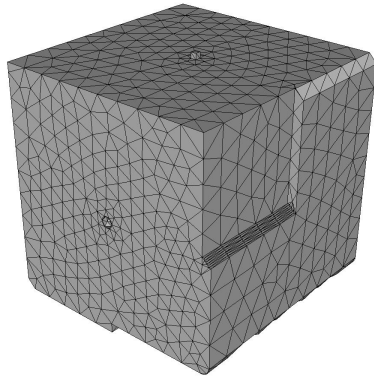


Fig. 1c. 27,181 tets with average element quality 0.8411 on original model

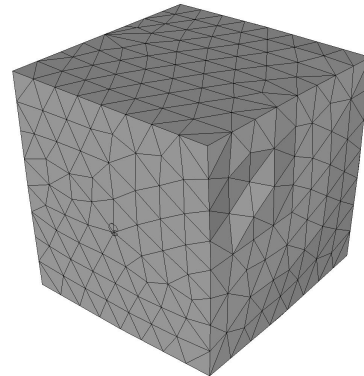


Fig. 1d. 3,879 tets with average element quality 0.8615 on defeatured model

Figure 1a shows a simple model with small features including multiple steps, chamfers, imprinted curves, pegs, and holes, which would make the hex meshing of this model difficult. Figure 1b shows a defeatured representation that has been reduced to a simple cube. In this case a hexahedral mesh can be generated on the model. Figure 1c shows the unreduced model with a tet mesh applied and the reduced model in Figure 1d also with a tet mesh. The number of tet elements without and with defeaturing are 27,181 and 3,879 respectively, significantly reducing the degrees of freedom in the mesh. The average element quality is also higher in the defeatured model as the small features, that would have required smaller element sizes, are no longer present - consequently avoiding large mesh size transitions.

It should be noted that the proposed approach does not directly address the CAD clean-up and repair issues. Instead we assume that the input CAD data is geometrically and topologically valid. While the focus of this work is on the defeaturing of correctly resolved solid models, the authors believe that the proposed framework can be extended to also address common CAD repair issues.

2 Background

Third party geometry kernels in CUBIT [4] such as ACIS, Granite, or Catia provide a rich set of geometry query and modification capabilities; however, they typically do not provide tools to automatically control defeaturing of CAD models for mesh generation. As a consequence, this work is intended to address the defeaturing of CAD models, particularly those that are generated in commercial 3D solid modelers (such as SolidWorks, Pro/Engineer, or UniGraphics) and are then imported into an independent mesh generation tool such as CUBIT. Built into many 3D solid modelers is a convenient feature or parameter-based representation of the model that provides capability to simply turn off unwanted features or modify parameters to simplify the model. It is clear that analysts should take advantage of parameter-based defeaturing whenever available prior to exporting to a third party mesh generation tool. In practice, however, the full design model containing features irrelevant to the simulation, is used as the basis for mesh generation. For the cases in which this work addresses, the model is provided without feature information and with only the boundary representation (B-Rep) topology and NURBS-based geometry definition. As a consequence, the users usually spend significant amount of time in manually identifying unwanted features and applying appropriate modification operations to simplify the model prior to mesh generation.

Techniques for feature suppression and defeaturing have been widely studied with many approaches presented in the literature. A recent study by Thakur et. al [14] surveys CAD model simplification techniques and classifies existing methods into four categories: techniques based on surface entity based operators, volume entity based operators, explicit feature based

operators, and dimension reduction operators. Based on Thakur's classification, surface-based operations are most applicable to use within the context of the required application. Techniques for surface-based defeaturing operations can be categorized based upon whether operations are applied directly to the continuous NURBS-based geometry description or whether they utilize discrete representation of the domain.

NURBS-based defeaturing, such as those described by Clark [3] have the advantage of accurately redefining the topology and geometry. While geometry kernels can assist by providing the necessary geometric operations, identifying the numerous topologic configurations where defeaturing should occur in a consistent and robust manner is an open-ended problem. Inoue et al. [10] reported a face clustering technique for FE mesh generation similar to virtual topology [1]. The approach iteratively merged the model faces to obtain face cluster regions having sufficiently large area compared to the mesh element size. Authors defined metrics for mesh area, boundary smoothness, and surface flatness and used them for ranking the faces for merging. This approach is efficient for 2-manifold surface models and does not defeature topological features like holes. Foucault et al. [8] proposed a hypergraph-based Mesh Constraint Topology (MCT) for simplification. Composite topological entities of MCT are defined as the union of Riemannian surfaces/curves constituting the reference model. Graph-based operators then perform delete, insert, collapse, and merge of MCT entities. Indeed, Thakur et al. [14] provides a comprehensive list of literature where authors have attempted to enumerate various cases of defeaturing directly on NURBS-based solid models.

Alternatively, several methods have been proposed that perform defeaturing in a discrete domain. For example, Gao et al. [9] use a feature recognition strategy to identify features for suppression in the continuous model. Following removal of the features a Delaunay triangulation approach is used to fill gaps or holes that may be left. Unfortunately, feature-based methods can often lack completeness as defeaturing only a subset of features, such as holes or fillets may miss other regions that are not identified as features such as narrow gaps or misalignments in an assembly. Fine et al. [6] used operators based on vertex removal and spherical error zone concept to transform the input polyhedral geometry while preserving it within an envelope. This envelope is obtained from a mechanical criterion which can be based either on an a posteriori error estimator or on a priori estimation.

Other discrete methods perform defeaturing as part of the meshing procedure or as a post-process to meshing. Mobley et al. [11] propose a method that does not directly modify the geometry, but rather defines operations within the surface meshing algorithm that ignores or combines features. Borouchaki and Laug [2] perform operations on the triangles of the mesh following the initial meshing operation of individual surfaces. Simplification is performed by identifying surfaces to be combined and then optimizing local mesh quality while ignoring the boundary between identified regions. Foucault et al. [7] extended the advancing front method to composite geometry. Dey et al. [5]

defined a priori error metrics based on element quality, and edge-collapse operations were iteratively performed on poor quality elements as a post-meshing operation.

Performing the reduction operations on the mesh itself can however adversely influence the mesh quality. If, for example sliver features are present in the model, mesh sizes in the initial mesh will be significantly finer than is needed in the final mesh. Aggressive collapse operations are then needed in order to achieve the desired size and quality once the slivers have been identified. In contrast, the proposed method performs reduction operations on a facet-based discrete model that represents geometry. This defeated discrete model will later be used as the underlying geometry for a meshing algorithm. The quality of the triangles in the facet representation is not critical provided the underlying geometry is reasonably represented. As a result, the reduction operations do not have to concentrate on maintaining a quality triangulation, as would be required with other methods that perform defeaturing during or after mesh generation.

Performing defeaturing on the mesh is however attractive because it leaves the original model untouched for subsequent meshing operations for alternate element resolutions or for application of boundary conditions. However, one of the drawbacks is that the identification of features to be suppressed as well as specific defeaturing operations must be integrated directly within the meshing algorithm. For CAD-based tools such as CUBIT, which includes numerous options for surface meshing, it would be advantageous to perform defeaturing operations independent of any specific meshing algorithm, allowing for any surface meshing procedure to utilize defeaturing without special modifications. Thus, our method provides the advantages of mesh-based defeaturing without the associated drawbacks.

Following a review of the various approaches for defeaturing, the following list summarizes the criteria for which our automatic defeaturing framework has been developed:

1. A strategy for identifying features for suppression should utilize only the topology and geometry of the B-Rep NURBS model and should not consider any additional data such as feature data.
2. The procedure should be applicable for any surface mesh generation algorithm, including paving, triangle advancing front and Delaunay methods. Specialized changes to individual meshing algorithms to accommodate defeaturing should be avoided.
3. The final mesh should maintain associativity with the original model. The attributes and boundary conditions applied on the original B-Rep model should be mapped to the final mesh.
4. The algorithm should not be based on any specific proprietary geometry kernel as multiple geometry kernels exist within the CUBIT environment.
5. Finally, the extent of automatic feature suppression must be user controllable. Individually identifying features for preservation or for suppression should also be provided.

The authors assert that the proposed defeaturing algorithm meets the criteria described above.

3 Defeating Algorithm

Here we briefly discuss the steps involved in defeaturing irrelevant features of a NURBS based B-Rep CAD model via a discrete model. A robust method for automatically detecting the irrelevant features is very critical to the success of the proposed approach. A geometry-based mesh sizing function is used as the field to automatically identify the irrelevant features that need to be defeated. With the proposed method, the original CAD model is not altered as the actual defeaturing is performed on a discrete model of the original CAD model. The following illustrates the basic steps used in the defeaturing algorithm.

Figure 2a shows a typical industrial CAD model containing common features such as holes, fillets, blends, and steps. Note that the zoom view shows a small step which is an unintended feature. Such a small step would result in poor element quality if retained. This model also contains multiple tiny holes. Assuming these features are not significant to the simulation, they can significantly increase the element and node count in the final mesh, resulting in increased analysis time and memory usage.

Figure 2b shows a discrete representation of the input CAD model. First, the facets of each surface are obtained and stitched to form a water tight model. The B-Rep topology is then embedded in the facet-based discrete model, establishing the associativity between facet entities and the original B-Rep topological entities.

Figures 2c and 2d show the defeated model represented via the facet-based B-Rep model. Note that the holes and steps are no longer seen in the defeated model. To accomplish defeaturing, two main steps are used: (1) Identification of irrelevant features on the discrete model; and (2) Performing facet-based reduction operations to remove irrelevant features.

Reduction operations are first performed on lower order entities followed by higher order. For example, the facet entities associated with curves are first collapsed before collapsing the facet entities associated with surfaces. Following each reduction operation, care is taken to ensure a valid B-Rep is maintained and that the mapping between the topological entities of the original CAD model and that of the reduced discrete model is updated. This is essential for decoding meshing attributes such as meshing schemes, size specifications and boundary conditions on the discrete model. Thus, the defeated model can be updated and meshed using the attributes defined on the original solid model.

Figures 2e and 2f show the mesh on the defeated and the original models, respectively. Mesh generators in CUBIT are capable of using the discrete B-Rep model as the basis for their geometry. As a result it is not currently necessary to convert the defeated model back into a NURBS B-Rep

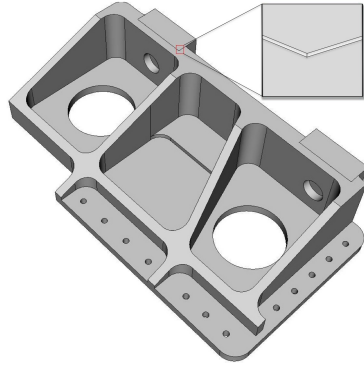


Fig. 2a. Original B-Rep CAD Model

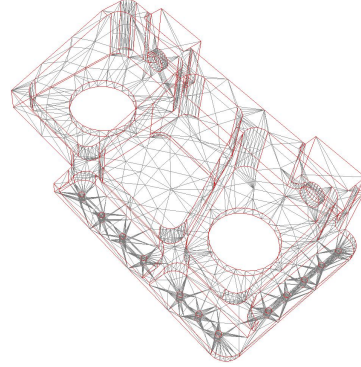


Fig. 2b. Discrete B-Rep Model

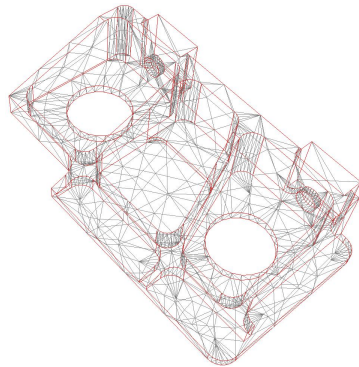


Fig. 2c. Defeatured Discrete Model

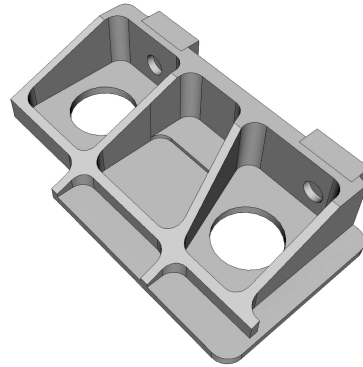


Fig. 2d. Defeatured Discrete Model

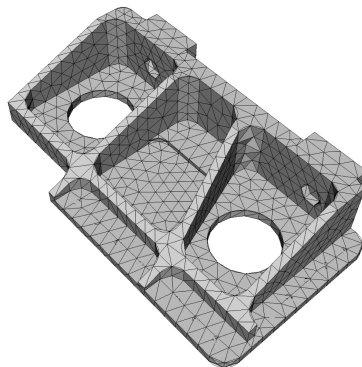


Fig. 2e. Tet mesh on Defeatured Model

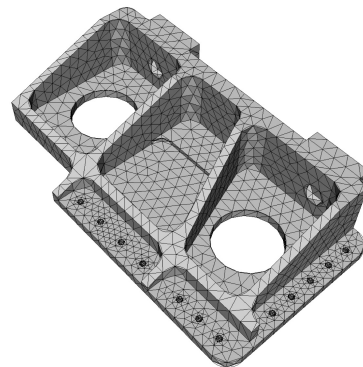


Fig. 2f. Tet mesh on Original Model

format if only the mesh is desired. Note that in Figure 2e that the fine mesh around the holes is no longer seen in the defeatured model. Also as the steps are defeatured the element quality improves significantly. The minimum tet

quality in Figure 2e is 0.038 compared to 0.004 in Figure 2f. Also note that the number of tet elements in the defeatured model is 4,975 which is much less than the 11,328 tet elements in the model with no defeaturing. As the mapping between the entities of the original and the defeatured models has been maintained, the mesh can be associated with either or both discrete and continuous models. We now discuss each of the steps of the defeaturing procedure in more detail.

3.1 Obtaining a Discrete Model

The first step of the procedure is to obtain a discrete representation of the CAD model. Requirements of the discrete model include the following:

1. The deviation of the facets from the original CAD model should be proportional to the user defined mesh size. For example, a small mesh size would require a proportionately small facet size so that curvature can adequately be captured in the final mesh.
2. The discrete representation should be watertight. This implies that each adjacent surface shares common facet edges with its neighbors.
3. Relationships should be maintained between the original B-rep entities and the discrete entities. This is to ensure that the final mesh will be correctly associated with the original CAD model once the procedure is complete.

In most cases it is convenient to obtain the discrete representation from a third party CAD kernel such as ACIS or Granite. Building a watertight faceted representation of the model may require additional operations to stitch adjacent surfaces to ensure surfaces share common edges. Additionally, to ensure correct associativity between the original B-Rep entities and the mesh entities that will be generated, the data to link back to the original B-Rep must be generated at this point and maintained throughout the procedure.

For the current work, the mesh-based geometry (MBG) [12] definition as proposed by Owen et al. [12] is used to build and represent the discrete representation once it is extracted from the CAD model. The MBG definition can also represent non-CAD-based models; for example models initially defined by triangle facets, such as STL formats, or by extracting geometry from a legacy FE mesh.

3.2 Detection of Features for Suppression

The identification of features for suppression, is a key component of the proposed method and is critical to the success of defeaturing. Feature identification is achieved using mesh sizing function work of the author [13]. This work describes a systematic approach to reveals the geometric factors that

completely capture the geometric complexity for controlling element sizes. Irrelevant features are defined as those entities where the mesh size determined via geometric factors falls below the user specified threshold value ε_* .

Below are the three main steps involved in detecting irrelevant features and are subsequently discussed:

1. Identify geometric factors of input solid model.
2. Measure geometric factors using a set of tools.
3. Mark irrelevant regions on the discrete model using the tools.

Identification of Geometric Factors

To begin the procedure, the input CAD model is first decomposed into disjoint subsets, defined as dimensionally-based groupings of geometric entities. Let an input CAD solid model S contain N surfaces, F_i , where $i = 1, 2, \dots, N$, which are curvature-continuous and do not intersect at the interior. We can then define S in terms of the sum of its disjoint subsets as:

$$S = in(S) + \sum_{n=1}^N in(F_n) + \sum_{m=1}^M in(C_m) + \sum_{l=1}^L V_l \quad (1)$$

where the disjoint subsets of solid S are, the interior of the solid, $in(S)$, the interior of each surface, $in(F_n)$, the interior of each curve, $in(C_m)$, and vertices, V_l . As the subsets are disjoint, the geometric complexity of each subset is independent of the others.

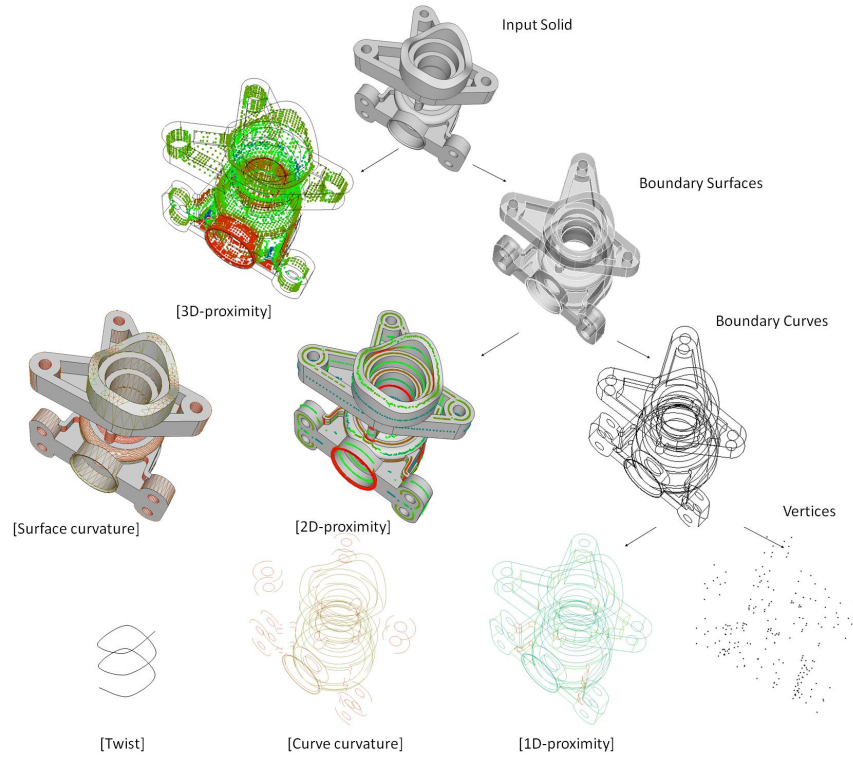
Table 1 tabulates the full list of geometric factors that are used to measure the complexity of each disjoint subset. Geometric factors identified on the disjoint subsets include 3D proximity, 2D proximity, surface curvature, curve curvature, 1D proximity, and curve twist. The user can optionally provide threshold values for the size field computed using these geometric factors. ε_{3D} , ε_{2D} , ε_{sc} , ε_{1D} , ε_{cc} and ε_t represent user defined threshold values for 3D proximity, 2D proximity, surface curvature, 1D proximity, curve curvature, and twist respectively. If not explicitly identified by the user, default values for ε_* are assigned.

Measuring Geometric Factors

The tools needed to measure the geometric factors of each disjoint subset are also shown in table 1 and figure 3. As shown in Figure 3, a 3D-skeleton and 2D-skeleton are used to measure 3D-proximity and 2D-proximity, respectively. 3D Skeletons, as described in [13], are computed using a grassfire approach that progressively marches over a PR-octree decomposition of the input solid model. A distance function which approximates the closest distance between an octree node and the boundary surfaces of the solid at each octree node, is updated as the grassfire is propagated from the octree nodes closest to the boundary towards the interior. The skeleton of the solid model

Table 1. Geometric factors, tools, and checks for disjoint subsets

Subset of CAD Model	Geometric Factors	Tools for Measuring Geometric Factor	Geometric Check
$in(S)$	3D proximity	3D skeleton distance (d_{3D})	$2d_{3D} < \varepsilon_{3D}$
$in(F_n)$	2D proximity	2D skeleton distance (d_{2D})	$2d_{2D} < \varepsilon_{2D}$
	surface curvature	min. principal radius of curvature (r_{min})	$r_{min} < \varepsilon_{sc}$
$in(C_m)$	1D proximity	curve length (l)	$l < \varepsilon_{1D}$
	curve curvature	radius of curvature (r_c)	$r_c < \varepsilon_{cc}$
	curve twist	torsion (t)	$f(t, r_c) < \varepsilon_t$

**Fig. 3.** Tools proposed to measure geometric factors of disjoint subsets

is approximated from the points that are generated where the opposing fronts collide. The distance function at the skeleton points can approximate one-half the local thickness or 3D-proximity between different combinations of geometric entities in a computationally efficient manner.

While an important geometric factor, the current implementation neglects 3D-proximity as only surface defeaturing is performed in the current work. As a result, only the geometric factors that capture the complexity of $in(F_n)$, $in(C_m)$, and V_l are considered. A more complete implementation should consider 3D-proximity to avoid inadvertently collapsing thin volumes that would otherwise not be detected with any other geometric factors. Future work may require a tetrahedral volumetric discrete representation to adequately perform volume defeaturing via tetrahedral collapse operations by considering 3D proximity. Currently, only surface defeaturing is performed using all the other geometric factors.

The 2D skeleton, also described in [13] measures 2D-proximity between boundary curves and vertices of a given surface. For a general surface the 2D-skeleton can be extracted in a similar manner to the 3D-skeleton by utilizing the same grassfire approach by progressively marching from the boundary curves towards the interior on the discrete model. To improve computational efficiency, on planar regions a chordal axis transform is used to extract the 2D-skeleton. Local edge swap operators are performed to remove illegal edges to obtain a more accurate 2D-skeleton. Figure 4b shows an example of the 2D-skeleton points extracted on multiple surfaces of the input solid. The red dots indicate where the smallest local thickness has been detected.

Other tools for measuring geometric factors, as illustrated in table 1, include minimum principal radius of curvature on a surface, r_{min} , curve length, l , radius of curvature of a curve, r_c and torsion, t . These values can be computed directly from the CAD solid model by querying the underlying geometry kernel, or else approximated from the discrete model. Since direct queries are computationally more expensive, approximating these values from the discrete representation is generally sufficient.

Identifying Features to be Suppressed

The geometric checks, as defined by the skeleton and other tools outlined in table 1 are used to identify irrelevant features for suppression. These tools calculate the maximum mesh size at a given point on a facet entity of the discrete model. A complete size field function $s = f(x, y, z)$ obtained by interpolating the distance at the skeleton points and other tools is shown in figure 4c with a color code. If the field value (maximum mesh size) is less than the minimum threshold value ε_* then that facet entity is marked for suppression. Note that the entities specifically identified by the user for preservation are ignored and never marked for suppression.

For example, unwanted features such as narrow regions and slender surfaces can be identified for reduction using the 2D-skeleton tool. In this case, the local thickness, or twice the skeleton distance ($2d_{2D}$) is used in calculating

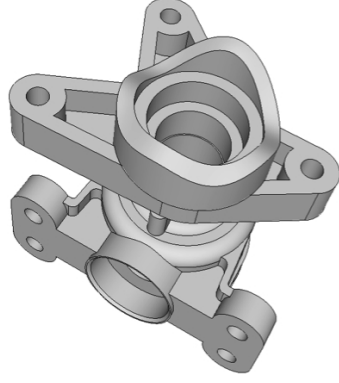


Fig. 4a. CAD model with narrow regions

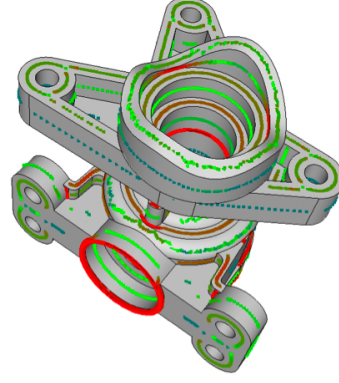


Fig. 4b. 2D-skeleton of each surface with color coded local thickness

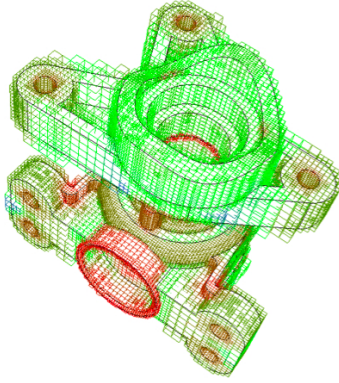


Fig. 4c. Mesh size based on skeleton and other tools

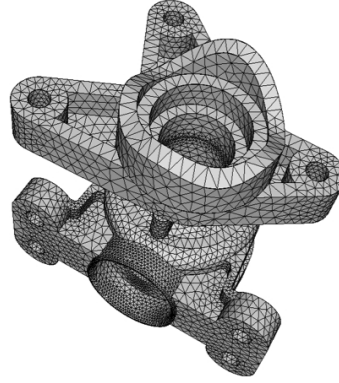


Fig. 4d. Fine mesh at skeleton points with smaller local thickness

maximum mesh size due to 2D-proximity. If the computed mesh size due to the 2D-skeleton is less than ε_{2D} then the facet entities associated with that skeleton point are marked for reduction. This ensures that by defeaturing these facets, mesh elements with sizes below the size ε_{2D} will not exist in the final mesh.

Similarly, other tools reveal the local maximum mesh size which in turn influences the detection of irrelevant features for a given ε . As the geometric checks are performed, the facet entities corresponding to the irrelevant features are detected and marked for suppression. For example, the 3D-skeleton tool, when used, can detect thin-walled regions via the 3D-proximity check; surface and curve curvature based checks can detect facet entities near high curvature regions such as fillets, blend patches, and holes; and 1D-proximity can detect facet edges associated with tiny curves and small features such as pegs and indentations via curve length checks.

3.3 Suppressing Features

Once all irrelevant features have been identified, operations to suppress the features can be performed. This is accomplished by using a standard edge collapse operator on all edges marked for suppression as illustrated in figure 5. To accomplish this, the topological entities of the discrete B-Rep model are visited in a dimension-based order. For example, facet edges associated with the curves are collapsed prior to collapsing those associated with the interior of the surface. As the collapse operations are performed the associativity between the geometric entities of the discrete model and the facet entities are updated. This may require that following any given edge collapse that the resulting edges may need to maintain associativity to multiple geometric entities. This generalized one-to-many, child-parent associativity between facet entity and original CAD B-Rep entity is maintained and updated throughout the procedure.

Following any operation, local checks and updates are made to the reduced facet-based B-Rep to ensure a valid topological configuration is always

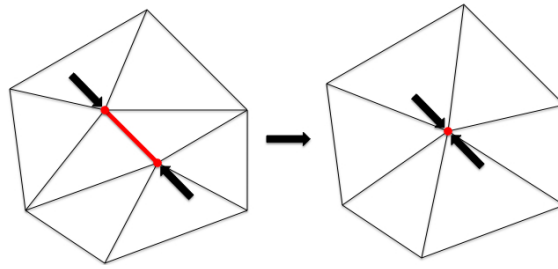


Fig. 5. The standard edge collapse operation

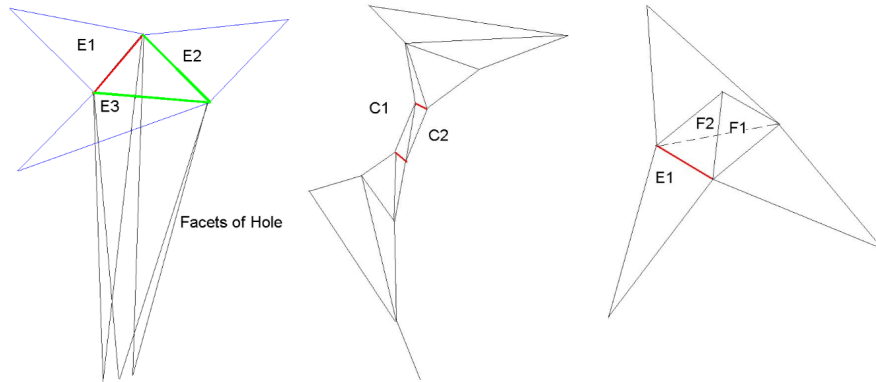


Fig. 6a. Single curve overlap

Fig. 6b. Two curves overlap

Fig. 6c. Single surface overlap

maintained. Figures 6a to 6c illustrate three specific cases that may make the discrete model invalid after an edge collapse operation.

Figure 6a shows an edge collapse at a cylindrical hole. In this case, the end circle of the cylindrical hole has been reduced to the three edges E1, E2, and E3, which are associated with one circular curve. After collapsing edge E1, the edges E2 and E3, although separate edges will share the same end vertices. To resolve this case, edges E2 and E3 are merged and represented as a single edge in the facet model. This ensures that there will be only one facet edge incident on any given pair of facet vertices.

Figure 6b shows that collapsing two small facet edges (shown in red) whose end vertices are incident on two curves C1 and C2 will result in a facet-edge associated with both the curves C1 and C2. This results in an invalid topology in the discrete B-Rep model even though the geometry continues to be watertight. To make the topology valid, both the curves C1 and C2 are split at their common overlapping edges to form a third curve C3. The facet edges at the overlapping region are then associated with a new curve C3 instead of associating with both C1 and C2. Although C3 is created as a new distinct curve entity in the discrete model, associativity back to its original geometry owners in the CAD B-Rep is maintained.

Figure 6c shows the facets of a single surface with edge E1 as one of the base edges of a triangular pyramid. Performing a collapse operation on edge E1, which is not an edge of F1 or F2 would cause the two facets F1 and F2 to overlap. Similar to the case described in figure 6a, although the facets are unique entities, they would share common vertices, creating a non-manifold or dangling facet. To avoid this case, the facets F1 and F2 are first merged and then subsequently deleted as part of the collapse operation of edge E1.

3.4 Meshing the Defeatured Model

Once all features identified for suppression have been reduced, the discrete model can be used as the basis for any of the surface and volumetric meshing tools in the CUBIT tool suite. Because associativity has been maintained throughout the reduction procedures, mapping between the original and the defeatured model is used to translate the user defined meshing schemes and mesh sizes prior to mesh generation. Since small features have now been removed from the discrete model, the resulting mesh in most cases will be higher quality with fewer degrees of freedom than if the model had not been defeatured. Using the same mapping between the original CAD model and discrete model, any boundary conditions specified in terms of the original CAD entities can be decoded onto the final FE mesh entities.

4 Results

The proposed approach has been implemented in C++ in CUBIT and has been tested on a limited number of industrial CAD models. We illustrate the effectiveness of the proposed method on two selected single part examples.

While this work focusses on single part models, work is underway to extend the framework to support volume defeaturing on assembly models.

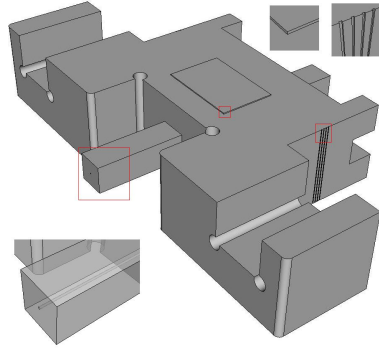


Fig. 7a. Original B-Rep CAD Model

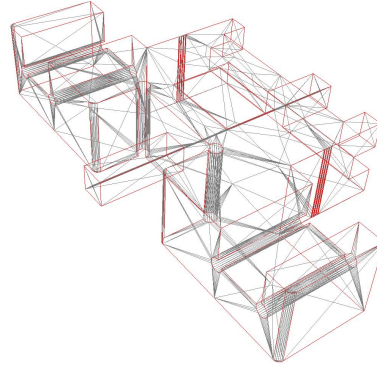


Fig. 7b. Discrete Model

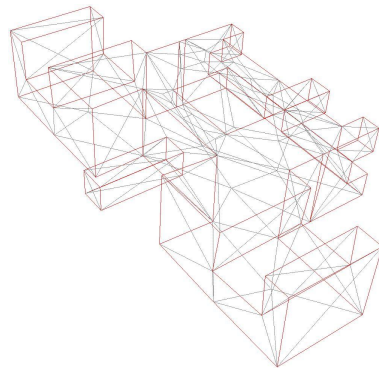


Fig. 7c. Defeatured Discrete Model

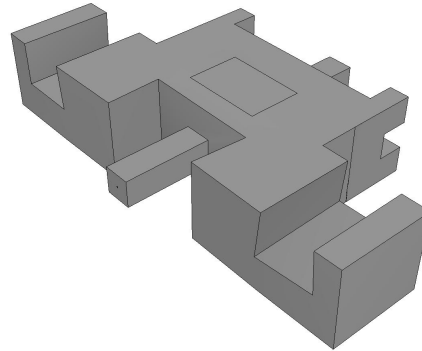


Fig. 7d. Defeatured B-Rep Model

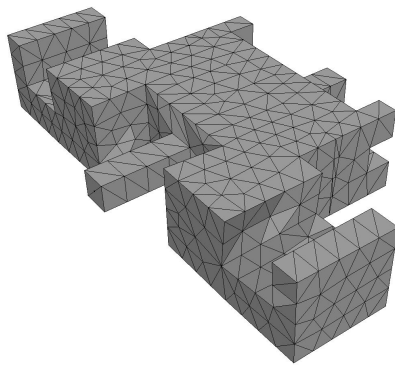


Fig. 7e. Tet mesh on Defeatured B-Rep Model

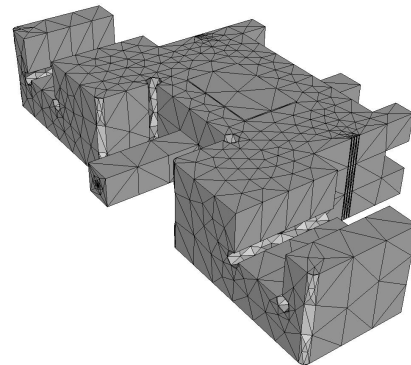
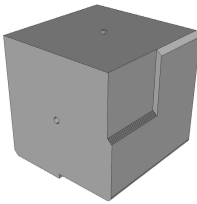
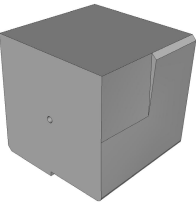
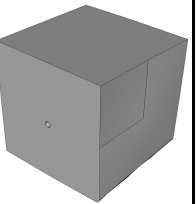
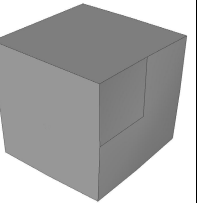
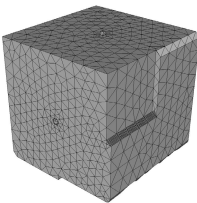
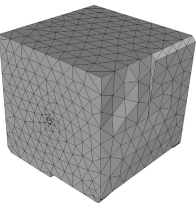
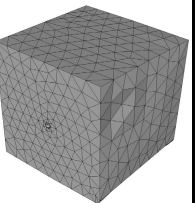
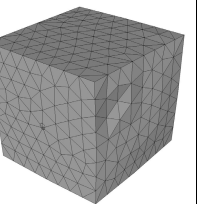


Fig. 7f. Tet mesh on Original B-Rep Model

Figures 7a to 7d show the different stages of defeaturing on a simple industrial part. Note that the model shown in Figure 7a and Figure 7b contains unintended features such as a long cylindrical hole, fillets at both convex and concave edges, multiple slots, and a protruded slab. Figures 7c and 7d show the defeatured model. Features have been removed based upon geometric factors that are controlled by a user specified threshold value of 1.0. Figure 7e shows a tetrahedral mesh on the defeatured model that has 3,490 tets with minimum element quality of 0.29 and average element quality of 0.825. In contrast, Figure 7f is a tetrahedral mesh on the model without defeaturing. The number of tet elements increases significantly to 51,109 and the minimum and average quality falls to 0.008 and 0.75, respectively. Both figures 7e and 7d use the same automatic triangle and tetrahedral meshing algorithms with the same default mesh settings.

Table 2 shows the effect of geometric factors on defeaturing. In CUBIT geometric factors can be controlled by issuing commands to enable a geometric factor. As more geometric factors are added, more facets get qualified for defeaturing and are marked for collapse. Note that using the geometric factors users can remove a specific type of feature. For example, adding the curve length check will remove the small peg at the top of the model. Subsequently adding the 2D-proximity criteria resulted in removing the chamfer. This is

Table 2. Controlling geometric factors to influence defeaturing

	No Features Removed	Small Curves Removed	Chamfer Removed	Hole & Fillet Removed
Geometric Model				
Mesh				
Geom. Factors	None	Curve Length	Curve Length 2D Proximity	Curve Length 2D Proximity Curve Curvature Surface Curvature

because the distance between the opposite long edges of the chamfer fell below the user specified threshold of 0.6. Similarly, adding curve and surface curvature-based checks removed the facets associated with the central hole and a fillet. In general, the geometric factors determine the mesh size at the facet entities first and then mark the facet entities for deletion only if the mesh size falls below the user specified threshold value.

5 Conclusion

This paper introduces a new facet-based approach to defeaturing CAD models by automatic identification of irrelevant features via a geometry-based size field. Robust detection of irrelevant features is achieved by using geometric factors that control the size field. The geometric factors are identified by a systematic analysis of the geometric complexity of the input CAD model. These geometric factors are also used as user controls to target the selection of irrelevant features, where the field value or mesh size, goes below a user specified threshold value. The actual defeaturing is performed on a facet-based discrete B-Rep model as it is less complex and more robust than performing it on the original NURBS model. It also has the advantage of leaving the original CAD model untouched so that multiple alternative defeatured representations can be derived based upon the needs of the simulation. As the meshing algorithms themselves are independent of the actual defeaturing procedures, any mesh generation scheme can be applied to the defeatured model without modification. The proposed method has been demonstrated on a limited set of industrial models and has proven effective in reducing user time, degrees of freedom, and mesh quality issues.

We have limited the initial work described here to feature suppression on single part CAD models. Extension of these procedures to assembly models will be a necessary next step. The proposed framework can also be extended to perform imprinting [15] and CAD repair via the discrete B-Rep model. Another potentially valuable area for study would be the introduction of physics-based sizing properties into the field function that drives the criteria for identification of features to be suppressed. This would provide a mechanism to automatically retain features in the model where important physics is occurring, but defeature where it is not.

References

1. Blacker, T.D., Sheffer, A., Clements, J., Bercovier, M.: Using Virtual Topology to Simplify the Mesh Generation Process. *Trends in Unstructured Mesh Generation*, vol. AMD-220, pp. 45–50 (1997)
2. Borouchaki, H., Laug, P.: Simplification of composite parametric surface meshes. *Engineering with Computers* 20, 176–183 (2004)
3. Clark, B.W.: Removing Small Features with Real CAD Operations. In: *Proceedings of the 16th International Meshing Roundtable*, pp. 183–198 (2007)

4. CUBIT Geometry and Mesh Generation Toolkit, Sandia National Laboratories (2009), <http://cubit.sandia.gov>
5. Dey, S., Shephard, M.S., Georges, M.K.: Elimination of the Adverse Effects of Small Model Features by the Local Modification of Automatically Generated Meshes. *Engineering with Computers* 13, 134–152 (1997)
6. Fine, L., Remondini, L., Leon, J.C.: Automated Generation of FEA Models Through Idealization Operators. *International Journal for Numerical Methods in Engineering* 49(1-2), 83–108 (2000)
7. Foucault, G., Cuilliere, J.C., Francois, V., Leon, J.C., Maranzana, R.: An Extension of the Advancing Front Method to Composite Geometry. In: *Proceedings of the 16th International Meshing Roundtable*, seattle (2007)
8. Foucault, G., Cuilliere, J.C., Francois, V., Leon, J.C., Maranzana, R.: Adaptation of CAD Model Topology for Finite Element Analysis. *Computer Aided Design* 40(2), 176–196 (2008)
9. Gao, S., Zhao, W., Yang, F., Chen, X.: Feature Suppression Based CAD Mesh Model Simplification. In: *Proceedings IEEE International Conference on Shape Modeling and Applications*, NY, June 4-6 (2008)
10. Inoue, K., Itoh, T., Yamada, A., Furuhashi, T., Shimada, K.: Face clustering of a large-scale CAD model for surface mesh generation. *Computer Aided Design* 33(3), 251–261 (2001)
11. Mobley, A.V., Carroll, M.P., Canann, S.A.: An Object Oriented Approach to Geometry Defeaturing for Finite Element Meshing. In: *Proceedings 7th International Meshing Roundtable*, pp. 547–563 (1998)
12. Owen, S.J., White, D.R.: Mesh-Based Geometry. *International Journal for Numerical Methods in Engineering* 58(2), 375–395 (2003)
13. Quadros, W.R.: A Computational Framework for Generating 3D Finite Element Mesh Sizing Function via Skeletons, Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA, U.S.A (2005)
14. Thakur, A., Banerjee, A.G., Gupta, S.K.: A Survey of CAD Model Simplification Techniques for Physics-based Simulation Applications. *Computer Aided Design* 41(2), 65–80 (2009)
15. White, D.R., Saigal, S., Owen, S.J.: An Imprint and Merge Algorithm Incorporating Geometric Tolerances for Conformal Meshing of Misaligned Assemblies. *International Journal for Numerical Methods in Engineering* 59, 1839–1860 (2004)